APPLICATION

FOR

UNITED STATES LETTERS PATENT

TITLE: DATABASE ACCESS WITH MULTILEVEL LOCK

APPLICANT: MIROSLAV CINA

CERTIFICATE OF MAILING BY EXPRESS MAIL

Express Mail Label No. EL983022377US

December 3, 2003

Date of Deposit

DATABASE ACCESS WITH MULTILEVEL LOCK

TECHNICAL FIELD

[0001] This invention relates to database access with multilevel lock.

BACKGROUND

[0002] In a parallel processing environment, a database can be shared by more than one process. For example, a database can contain records, each record containing entries related to customers of a company and entries related to products that the customers have ordered. A first process related to customer relationship management can attempt to access the database to update customer contact information of a particular record. A second process related to inventory control can also try to update a product specification contained in the same record. The customer relationship management information and the inventory control information within the same record can be dependent or independent of each other. To prevent multiple processes from updating the same record resulting in errors, a locking mechanism is provided so that before the first process attempts to modify a record, the first process "locks" the record to prevent other processes from modifying the record. When the second process attempts to modify a locked record, the attempt is rejected, and an error message is returned to the second process. The second process waits a period of time before attempting to modify the record again.

SUMMARY

[0003] In general, in one aspect, the invention features a method of scheduling access of a table by multiple processes, including associating a lock level with a particular process, a higher lock level representing a larger number of other processes having priority over the particular process in accessing the table; repeatedly attempting to associate the particular process with a lower lock level, and if the particular process has been successfully associated with the lower lock level, releasing a previous lock level associated with the particular process so that the previous lock level may be associated with other processes; and allowing the particular process to access the table when the lock level for the particular process is equal to a preset value.

[0004] This and other aspects of the invention may include one or more of the following features. The preset value is equal to one. Each of the processes attempts to associate itself with a lower lock level independently of other processes.

[0005] In general, in another aspect, the invention features a method that includes, upon receiving a request from a first process to access a record in a database, associating a first lock level to the first process and allowing the first process to access the record but preventing other processes from accessing the record until the first process finishes accessing the record; and upon receiving a request from a second process to access the record while the first process is still accessing the record, associating a second lock level to the second process. The method includes, when the first process finishes accessing the record, releasing the first lock level from being associated with the first process, and either (a) releasing the second lock level from being associated with the second process and associating the first lock level with the second process, allowing the second process to access the record but preventing other processes from accessing the record until the second process finishes accessing the record, and when the second process finishes accessing the record, releasing the first lock level from being associated with the second process, or (b) releasing a lock level from being associated with a third process and associating the first lock level with the third process, allowing the third process to access the record but preventing other processes from accessing the record until the third process finishes accessing the record, and when the third process finishes accessing the record, releasing the first lock level from being associated with the third process.

[0006] This and other aspects of the invention may include one or more of the following features. Preventing other processes from accessing the record includes allowing the other processes to read the record but not modify the record. The record is locked when the first lock level is associated with a process. Information is written to a queue to specify which lock level is associated with which process. At least two of the first, second, and third processes are run in a parallel processing environment.

[0007] In general, in another aspect, the invention features a method that includes locking a record in a database at multiple levels when multiple processes running in parallel attempt to access the record; assigning a lock level to each of the multiple processes, different processes

having different lock levels; and selectively permitting one of the multiple processes to access the record at a time.

[0008] This and other aspects of the invention may include one or more of the following features. The lock levels of the processes are reassigned when a process accessing the record terminates its access to the record. A process that attempted to access the record earlier than another process is assigned a lower lock level than the other process, and each process other than the process terminating its access to the record is assigned a lower lock level when the process terminates its access to the record. Information is stored in a queue indicating which process is associated with which lock level. Multiple instances of a procedure are called, where the procedure assigns a lock level to a process, and each instance of the procedure being associated with one of the multiple processes and configured to attempt to assign a different lock level to the process until the process is granted access to the record.

[0009] In general, in another aspect, the invention features a system that includes a database to store records, and a queue to store information relating to lock levels of processes that attempt to access the records, different processes having different lock levels when accessing the same record, one of the processes having a particular lock level being allowed to access the record.

[0010] This and other aspects of the invention may include one or more of the following features. The system includes a memory to store software code for implementing a procedure in which instances of the procedure are used to assign lock levels to the processes. The software code is configured so that the instances of the procedure are run in parallel.

[0011] In general, in another aspect, the invention features a computer program product, tangibly stored on a machine-readable medium, for implementing a multi-level lock process, comprising instructions operable to cause one or more programmable processors to lock a record in a database at a first level to allow a first process to modify the record but prevent other processes from modifying the record; and lock the record at a second level to allow a second process to modify the record after the record is unlocked at the first level but to prevent other processes from modifying the record while the record is being modified by the second process.

[0012] This and other aspects of the invention may include one or more of the following features. The computer program product includes instructions operable to cause the one or more programmable processors to, after the record is locked at the second level, check whether the

record is locked at the first level, and if the record is not locked at the first level, lock the record at the first level a second time and unlocking the record at the second level. The computer program product includes instructions operable to cause the one or more programmable processors to write to a queue to specify that the record has been locked at the first level and/or the second level. The computer program product includes instructions operable to cause the one or more programmable processors to run the first and second processes in parallel.

[0013] In general, in another aspect, the invention features a computer program product, tangibly stored on a machine-readable medium, for implementing a multi-level lock process, includes instructions operable to cause one or more programmable processors to, upon receiving a request from a first process to access a record in a database, associate a first lock level to the first process and allow the first process to access the record but prevent other processes from accessing the record until the first process finishes accessing the record; and upon receiving a request from a second process to access the record while the first process is still accessing the record, associate a second lock level to the second process. The computer program product includes instructions operable to cause one or more programmable processors to, when the first process finishes accessing the record, release the first lock level from being associated with the first process, and either (a) release the second lock level from being associated with the second process and associate the first lock level with the second process, allowing the second process to access the record but preventing other processes from accessing the record until the second process finishes accessing the record, and when the second process finishes accessing the record, releasing the first lock level from being associated with the second process, or (b) release a lock level from being associated with a third process and associate the first lock level with the third process, allowing the third process to access the record but preventing other processes from accessing the record until the third process finishes accessing the record, and when the third process finishes accessing the record, releasing the first lock level from being associated with the third process.

[0014] This and other aspects of the invention may include one or more of the following features. The instructions cause the one or more programmable processors to allow the other processes to read the record but not modify the record. The computer program product includes instructions operable to cause the one or more programmable processors to lock the record when the first lock level is associated with a process. The computer program product includes

instructions operable to cause the one or more programmable processors to write to a queue to specify which lock level is associated with which process. The instructions cause the one or more programmable processors to execute at least two of the first, second, and third processes in a parallel processing environment.

[0015] In general, in another aspect, the invention features a computer program product, tangibly stored on a machine-readable medium, for scheduling access of a table by multiple processes, including instructions operable to cause one or more programmable processors to associate a lock level with a particular process, a higher lock level representing a larger number of other processes having priority over the particular process in accessing the table; repeatedly attempt to associate the particular process with a lower lock level, and if the particular process has been successfully associated with the lower lock level, release a previous lock level associated with the particular process so that the previous lock level may be associated with other processes; and allow the particular process to access the table when the lock level for the particular process is equal to a preset value.

[0016] This and other aspects of the invention may include one or more of the following features. The instructions are configured so that the preset value is equal to one. The instructions are configured so that each of the processes attempts to associate itself with a lower lock level independently of other processes.

[0017] In general, in another aspect, the invention features a computer program product, tangibly stored on a machine-readable medium, for implementing a multi-level lock process, comprising instructions operable to cause one or more programmable processors to lock a record in a database at multiple levels when multiple processes running in parallel attempt to access the record; assign a lock level to each process, different processes having different lock levels; and selectively allow one of the multiple processes to access the record at a time.

[0018] This and other aspects of the invention may include one or more of the following features. The computer program product includes instructions operable to cause the one or more programmable processors to reassign the lock levels of the processes when a process accessing the record terminates its access to the record. The instructions are configured so that a process that attempted to access the record earlier than another process is assigned a lower lock level than the other process, and each process other than the process terminating its access to the

record is assigned a lower lock level when the process terminates its access to the record. The computer program product includes instructions operable to cause the one or more programmable processors to store in a queue information indicating which process is associated with which lock level. The computer program product includes instructions operable to cause the one or more programmable processors to call multiple instances of a procedure that assigns a lock level to a process, each instance of the procedure associated with one of the multiple processes and is configured to attempt to assign a different lock level to the process until the process is granted access to the record.

[0019] An advantage of the invention is that multiple users can send requests for accessing a shared database record even when another user is currently using the shared database record. The requests are automatically pipelined in a queue so that the users can access the shared database in sequence without being rejected or causing sharing violation errors.

[0020] Other features and advantages of the invention are apparent from the following description, and from the claims.

DESCRIPTION OF DRAWINGS

[0021] FIG. 1 is a diagram of a parallel processing environment that implements a multi-level lock process.

[0022] FIG. 2 is a flowchart of a multi-level lock process.

[0023] FIG. 3 is timing diagram of a multi-level lock process.

[0024] Like reference symbols in the various drawings indicate like elements.

DETAILED DESCRIPTION

[0025] As shown in FIG. 1, in a parallel processing system 80, multiple software processes (e.g., processes A, B, and C) access and process a table 84 in a database 82. System 80 implements a multilevel lock procedure 100 (FIG. 2) to allow the processes to share table 84 without causing errors when the processes A, B, C attempt to access the same table 84 at the same time. Each process A, B, C attempting to access a table is assigned a "lock level." A process (A, B, or C) is assigned a lock level equal to n if previous (n-1) lock levels have previously been assigned to other processes currently using the same table 84 or waiting to use

the table 84. Each process A, B, C attempts to obtain a lower lock level, and if successful, unlocks (releases) the previous lock level so the previous lock level can be available to other processes. A process (A, B, or C) is granted access to the table 84 when its associated lock level is equal to 1. The multilevel lock procedure 100 (described below) provides an efficient scheduling mechanism to allow multiple processes running in parallel to access and process data in table 84.

[0026] For example, initially, when process A attempts to access (86) table 84, process A is assigned a "lock level" that is equal to 1 since no other processes are accessing table 84. The assignment of lock level 1 to process A is stored in a first-in-first-out (FIFO) queue 88. Table 84 is locked (90), and process A is granted access to table 84. While process A is processing data in table 84, process B attempts (92) to access the same table 84. Process B is assigned a lock level that is equal to 2 since lock level 1 is already assigned to process A. The assignment of lock level 2 to process B is stored in FIFO queue 88. Before process A finishes using table 84, process C attempts (94) to access the same table 84. Process C is assigned a lock level that is equal to 3 since lock levels 1 and 2 have already been assigned to other processes. The assignment of lock level 3 to process C is stored in FIFO queue 88. When process A successfully executes and finishes using table 84, lock level 1 is released (98), and the table 84 is unlocked (99). This allows process B to access table 84.

[0027] The multilevel lock procedure 100 can be implemented as, e.g., a software process or an application programming interface (API), that is called by other processes when accessing a table in database 82. In the description below, depending on context, the term "procedure" is used to refer to the procedure itself or the software (e.g., process or API) that implements the procedure. Depending on context, the term "process" can be a noun (as in "a process that accesses the data in the database) or a verb (as in "to process the data in the database").

[0028] As shown in FIG. 2, procedure 100 starts (102) by assigning (104) a variable n to be equal to 1. The variable n represents a lock level that is assigned to a process (e.g., process A). An attempt is made to assign (106) a lock level n to the process, and a determination (108) is made whether lock level n is successfully assigned. Information relating to which lock levels have been assigned to which processes are stored in FIFO queue 88. If lock level n cannot be successfully assigned to the process, n is incremented (110) by 1, and the determination (106 and

108) of whether lock level n can be assigned to the process is repeated. If lock level n can be successfully assigned to the process, a determination (112) is made whether n is equal to 1. If n is not equal to 1, then n is decremented (128) by 1.

[0029] Procedure 100 attempts (130) to assign lock level n to the process, and determines (132) whether the lock level can be successfully assigned to the process. If lock level n can not be successfully assigned to the process, the attempt (130) to assign lock level n to the process is repeated. Procedure 100 can pause for a predetermined period between attempts to assign lock level n to the process. When the lock level n is successfully assigned to the process, the lock level (n+1) is unlocked (134); i.e., the assignment of lock level (n+1) is released from the queue so that lock level (n+1) is available to other processes.

[0030] The determination (112) is made whether n is equal to 1. If n is equal to 1, an attempt to lock (114) the table 84 is performed. Locking the table 84 can be achieved by calling a locking API provided by system 80. Locking the table 84 allows one process to access the table and prevents other processes from accessing the table, or allows other users to read but not modify the table. A determination (116) is made as to whether the table 84 is successfully locked. If the table 84 is not locked, procedure 100 ends and returns an error message (118). If the table 84 is locked, access to table 84 is granted to the process, and the process can process (120) and modify the data in table 84. After the process finishes processing and updating table 84, the process calls an unlock API provided by system 80 to unlock (122) table 84. Procedure 100 unlocks (124) lock level 1 (i.e., releases lock level 1 from the FIFO queue 88 so that lock level 1 can be assigned to others), and ends (126).

[0031] Each time a process attempts to access a table 84 in database 82, the process calls procedure 100. When multiple processes attempt to access tables in database 82, multiple instances of the procedure 100 are executed in parallel to schedule access of the tables. The following describes how processes A, B, and C access the table 84 by calling procedures 100a, 100b, and 100c (which are instances of procedure 100), respectively, to schedule access of the table.

[0032] As shown in FIG. 3, a timing diagram 200 shows events that happen when processes A, B, and C attempt to access table 84. Process A requests (202) access to table 84 by calling procedure 100a. Procedure 100a determines that the table 84 is not locked, assigns (204) a lock

level 1 to the process A, locks (206) table 84, and grants (208) access of the table 84 to process A. While process A is processing (210) data in table 84, process B requests (212) access to table 84 by calling procedure 100b. Procedure 100b determines that lock level 1 has already been assigned to some other process, assigns (214) lock level 2 to the process B, and waits for process A to finish using table 84. While process A is using table 84 and process B is waiting for process A to finish, process C requests (216) access of table 84 by calling procedure 100c. Procedure 100c determines that the lock level 2 has already been assigned to some other process, assigns (218) lock level 3 to the process C, and waits for processes A and B to finish using the table 84.

[0033] Procedure 100b repeatedly tests whether it can assign lock level 1 to process B (130 and 132 in FIG. 2). Procedure 100c repeatedly tests whether it can assign lock level 2 to process C. When process A notifies (220) procedure 100a that process A has finished using table 84, procedure 100a unlocks level 1 (222) (i.e., makes lock level 1 available to other processes), and unlocks (224) table 84. Procedure 100b determines that lock level 1 is available, assigns (226) lock level 1 to process B, unlocks level 2, locks (228) table 84, and grants (230) access of table 84 to process B. Process B processes (232) data in table 84. When procedure 100c detects that level 2 is available, procedure 100c assigns (229) lock level 2 to process C and unlocks level 3.

[0034] Procedure 100c repeatedly tests whether it can assign lock level 1 to process C. When process B notifies (234) that process B has finished using table 84, procedure 100b unlocks (236) level 1 and unlocks (238) table 84. Procedure 100c determines that lock level 1 is available, assigns (240) lock level 1 to the process C, unlocks lock level 2, locks (242) table 84, and grants (244) access of table 84 to process C. Process C processes (246) data in table 84. When the process C notifies (248) procedure 100c that process C has finished using table 84, procedure 100c unlocks (250) level 1 and unlocks (252) table 84.

[0035] Other embodiments are within the scope of the following claims For example, each lock level may be associated with locking one record in one table, or locking multiple records in one table, or locking multiple records in multiple tables, or locking an entire table, or locking multiple tables.